INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2021, VOL. 67, NO. 2, PP. 289-294
Manuscript received September 13, 2020; revised April, 2021 DOI: 10.24425/ijet.2021.135978

An Enhanced IEEE1588 Clock Synchronization for Link Delays Based on a System-on-Chip Platform

Xiaohan Wei, Xingzhong Xiong, Zhongqiang Luo, Jianwu Wang, and Kaixing Cheng

***Abstract*—The clock synchronization is considered as a key technology in the time-sensitive networking (TSN) of 5G fronthaul. This paper proposes a clock synchronization enhancement method to optimize the link delays, in order to improve synchronization accuracy. First, all the synchronization dates are filtered twice to get the good calculation results in the processor, and then FPGA adjust the timer on the slave side to complete clock synchronization. This method is implemented by Xilinx Zynq UltraScale+ MPSoC (multiprocessor system-on-chip), using FPGA+ARM software and hardware co-design platform. The master and slave output Pulse Per-Second signals (PPS). The synchronization accuracy was evaluated by measuring the time offset between PPS signals. Contraposing the TSN, this paper compares the performance of the proposed scheme with some previous methods to show the efficacy of the proposed work. The results show that the slave clock of proposed method is synchronized with the master clock, leading to better robustness and significant improvement in accuracy, with time offset within the range of 40 nanoseconds. This method can be applied to the time synchronization of the 5G open fronthaul network and meets some special service needs in 5G communication.**

***Keywords*—IEEE1588, clock synchronization, fronthaul, TSN, System-on-chip**

I. INTRODUCTION

THE TSN technology plays an important role in the field of 5G fronthaul. The task of TSN is to ensure that data packets are delivered within a certain time window without loss or high delay.

In IP based network, putting forward higher requirements for the network clock of data communication network equipment[1,2]. Traditional NTP cannot efficiently and timely transmit some special messages. And the increasing demand for network functions by the public means that the communication network needs to achieve higher accuracy. NTP can only achieve the synchronization accuracy of the millisecond level, far from meeting the synchronization needs of 5G for time synchronization standard. The emergence of IEEE1588 PTP can replace NTP. NTP obtains the timestamp at the application layer, while PTP can obtain the timestamp at the physical layer with the assistance of hardware, which greatly reduces the jitter and delay of packet transmission in the protocol stack [3,4,5]. However, in the IEEE1588v2 protocol, the communication link from the master-to-slave and the slave-to-master is generally considered to be symmetrical $delay = D_{UL} = D_{DL}$. Although the *delayAsymmetry*, which is a variable to compensate for the link has been introduced in IEEE1588v2, the specific calculation process is not clear [6]. In telecommunication networks, many links are asymmetric. Therefore, if the calculation continues according to the agreement and the delay of the upper and lower links is equal, it will inevitably cause errors [7].

In some past studies, Reference [8] chose ARM as the main control chip with the physical layer chip to achieve time stamping in the physical layer. The author also explained some functions in the engineering implementation. The implementation can finally achieve synchronization accuracy of nanosecond level. Still the author did not consider the impact of uplink and downlink asymmetry on clock accuracy. Based on the state transition model of the clock [9], established a clock servo system based on Kalman filtering. Simulated the impact of clock adjustment behavior and the changes of parameters on its performance. However, it does not take into account the link asymmetry in the multi-hop network. Reference [10] used an algorithm called block burst to calculate the asymmetry coefficient of an asymmetric link, which solved the problem of the compensation value of the asymmetric link. However, this method increased the network traffic overhead. According to the above-mentioned problems, this paper proposed a method that can optimize the value used to adjust the slave clock timer. The algorithm contains three steps. Firstly set the stable state in the program, getting the first threshold through the stable state. Secondly, by analyzing the distribution of the asymmetry ratio, the ratio of the good data can be obtained. This data is helpful for setting the second threshold. In the next part, A multi-level

This work was supported in part by the Natural Science Foundation of China under Grant 61801319, in part by Sichuan Science and Technology Program under Grant 2020JDJQ0061 and 2021YFG0099, in part by the Education Agency Project of Sichuan Province under Grant 18ZB0419, in part by the Opening Project of Key Laboratory of Higher Education of Sichuan Province for Enterprise Informationalization and Internet of Things under Grant 2017WZJ01, in part by the Major Frontier Project of Science and Technology Plan of Sichuan Province under Grant 2018JY0512, in part by Sichuan Science and Technology Program under Grant 2019YJ0476, in part by Outstanding Young Science and Technology Talent Project of Sichuan Provincial Department of Science and Technology under Grant 2020JDJQ0075, in part by the Sichuan Institute of Technology Graduate Innovation Foundation under Grant Y2019016, in part by Sichuan University of Science and Engineering Talent Introduction Project under Grant 2020RC33.

Xiaohan Wei, Xingzhong Xiong, Zhongqiang Luo, Jianwu Wang, and Kaixing Cheng are with School of Automation and Information Engineering and Artificial Intelligence Key Laboratory of Sichuan Province, Sichuan University of Science and Engineering, Yibin, China(e-mail: weixiaohans@163.com, xzxiong@suse.edu.cn,luozhongqiang@suse.edu.cn,wangjianwuradar@163.com, C1605221321@163.com)

filtering device is set to ensure that the data used to adjust the slave clock is a valid data. Finally, through experiments, these relatively a few data are enough to meet the high time synchronization accuracy of the system. The only software solution for the PTP is easy to implement, but the accuracy of time synchronization will be deteriorated. Because the timestamps are taken at the application layer. The operating system and protocol stack will bring large delays. On the other hand, the hardware solution means that FPGAs has the ability to mark the timestamps at nearly the physical layer. Hence, the timestamps with hardware assistance are much more precise than the software timestamps. Recently the FPGA has evolved to SoC technology [17]. The SoC is an integrated circuit, which integrates processing units, memory, input/output ports and reconfigurable logic.

In this article, a multi-level filter was proposed to optimize the data and SoC designs to implement the modified PTP is compared with some previous methods. Firstly, Section II introduces the background of PTP. Secondly, in Section III, The proposed algorithm has been described. Section IV depicts the software designs and the topologies of the SoC. Then Shows the results of the experiment. Sections V concludes the proposed work.

II. Background and Error Analysis

In the 5G network architecture, the 5G fronthaul network uses different hardware and network protocols from the previous generation network. The Enhanced Common Public Radio Interface (eCPRI) interface replaces the CPRI (Common Public Radio Interface) dedicated interface. The 5G Ethernet fronthaul network can use eCPRI to carry control plane and data plane messages, and at the same time rely on PTP to achieve timing synchronization of the entire network.

There are two types of synchronization mechanisms, named the Delay request-response mechanism and the Peer delay mechanism. Only the Delay request-response mechanism is introduced here. The synchronization process of its mechanism is divided into one-step and two-step modes [11,12,13]. Here the author chooses to adopt a two-step approach. Because the one-step method means that the master clock must dynamically modify the timestamp in the synchronization message. This requires that the timestamp t1 is available when the synchronization message is encoded, and it takes a certain time to add t1 to the *Sync* message. Due to this delay, it also limits the Ethernet connection under 10 Gigabit. In Fig. 1 the specific synchronization steps are: initialize the network, select 2 common clocks, and determine them as the master clock and the slave clock respectively; then synchronize the clock between the master and slave clocks. The master clock first sends a *Sync* message, and the timestamp t1 is stamped on the master, and then the t1 information is placed in the *Follow_Up* message and sent to the slave clock. When the *Sync* message arrives at the slave, the timestamp t2 is stamped from the slave clock, and then the timestamp t1 is obtained when the *Follow_Up* message arrives. At this time, the slave clock needs to send a delay request message to the master clock, and at the same time record the timestamp t3 of the *Delay_Req* message sent by the slave clock. The master clock receives a *Delay_Req* message and sends a *Delay_Resp* message to the slave clock. *Delay_Resp* message, which contains the timestamp t4 of the master clock. According to the obtained timestamps t1, t2, t3, and t4, the transmission delay D_{UL} from master clock to slave clock and the transmission delay D_{DL} from slave clock to master clock can be shown in the equation(1) and equation(2).The time offset between the master and slave clocks can be calculated in the equation(3).

$$D_{DL} + Offset = t_2 - t_1 \tag{1}$$

$$Offset - D_{UL} = t_3 - t_4 \tag{2}$$

$$Offset = \frac{t_2 - t_1 + t_3 - t_4}{2} + \frac{D_{UL} - D_{DL}}{2} \tag{3}$$

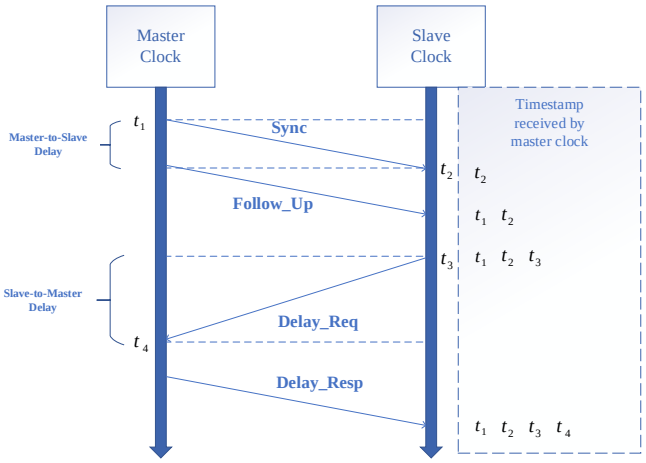

Fig. 1. Principle of IEEE1588 clock synchronization

IEEE officials generally believe that the IEEE 1588v2 clock-synchronized communication link is symmetrical, that is, $D_{UL} = D_{DL}$. Then the calculation method of time offset will become in the equation(4).

$$Offset = \frac{t_2 - t_1 + t_3 - t_4}{2} \tag{4}$$

Equation (4) is the final formula for the slave clock to calculate the time offset between the master and slave clock. After each master-slave synchronization message interaction completed, the slave clock substitutes the obtained four timestamps into equation (4) to obtain the time offset. Then, adjust the slave clock according to this time offset to achieve time synchronization [14]. Hence, the variation of the D_{UL} and D_{DL} will affect the offset.

III. An Enhanced Algorithm For Time Synchronization

Reference [15] pointed out that the uplink delay and the downlink delay change with the size of the message transmission and the link data rate. The way to calculate the delay is in equation (5).

$$D_{UL} = \frac{B_{sync}}{R_{DL}}, \quad D_{DL} = \frac{B_{Delay\_Req}}{R_{UL}} \tag{5}$$

B_{sync} and B_{Dealy_Req} are the sizes of Sync message and Delay_Req message. R_{DL}, R_{UL} are the downlink and uplink data rates of the link. From the result of equation(6), the asymmetry ratio R of the communication link can be calculated.

$$R = \frac{D_{UL}}{D_{DL}} \tag{6}$$

When running the traditional PTP algorithm, that is, simply think the uplink and downlink delays are equal. There is jitter during the transmission of network packet data. Through research, it is found that the value of offsets are affected by R. When the value of R is not equal to one, the traditional algorithm still thinks that the uplink and downlink delays are symmetrical. Hence, the offset calculated by it must have errors.

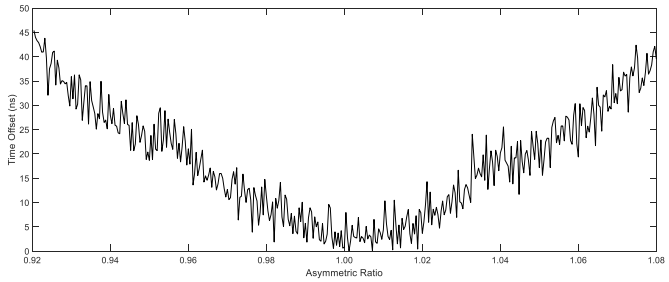

Fig. 2. Relationship between Asymmetric Ratio and Offset

Therefore, this paper proposes an algorithm that uses twice stage filtering to ensure the reliability of the samples through the synchronization system. Figure 4 shows the flowchart of the algorithm. First, let the system run the program. Since the system is a distributed network communication system that supports multicast messages, when the slave clock finds its master clock through an algorithm, the system is considered to be in a stable state. When system is in the stable state, set a value window in order to get the RMS(Root Mean Square) in real-time. The window's length *n* is a positive integer, as the time of the window, which is set according to the stability of the system. If the system is not particularly stable, the value of *n* can be increased appropriately. And then the first critical value has been defined. If the next sample Offset_n exceeds the critical value, the data is filtered out. Figure 2 shows the ralationship between offset and the asymmetric ratio. The value of offset has been absolute. When the asymmetry ratio is equal to 1, the offset from the master is almost zero. As shown in Fig. 3, approximately 41% of synchronized sample data have *R* between 0.95 and 1.05. Hence, 41% datas are good enough to adjust the slave clock. The boundary conditions *R* are definded between 0.95 and 1.05. When a new sample whose *R* exceeds the set boundary conditions, the sample will be filtered out. The synchronization interval of this calculation is 1 second.

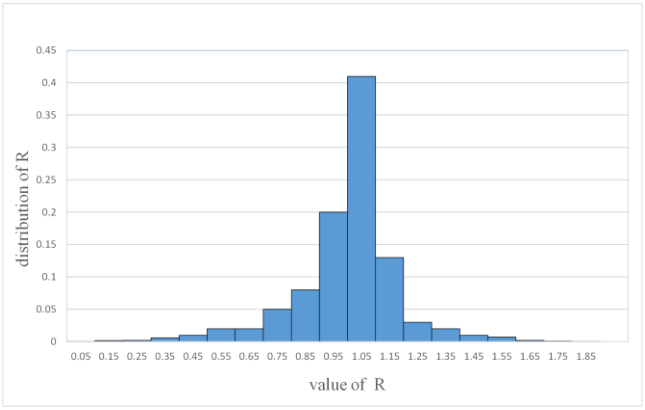

Fig. 3. Distribution of R in statistical samples

The calculation of RMS is shown in Equation (7). The value of *n* depends on the stability of the system. This data can be set manually. Offset_n is the newly input offset data, and Offset_o is the last offset data. Stable state means that the slave clock will no longer be adjusted sharply, but the servo system will calculate the clock offset compensation based on the input offset value. The unit of the compensation value is generally parts per billion. From now, the slave clock will be adjusted step by step and the state is stable. The offset may be positive or negative, because the clock may be faster or slower, it needs to be processed with an absolute value.

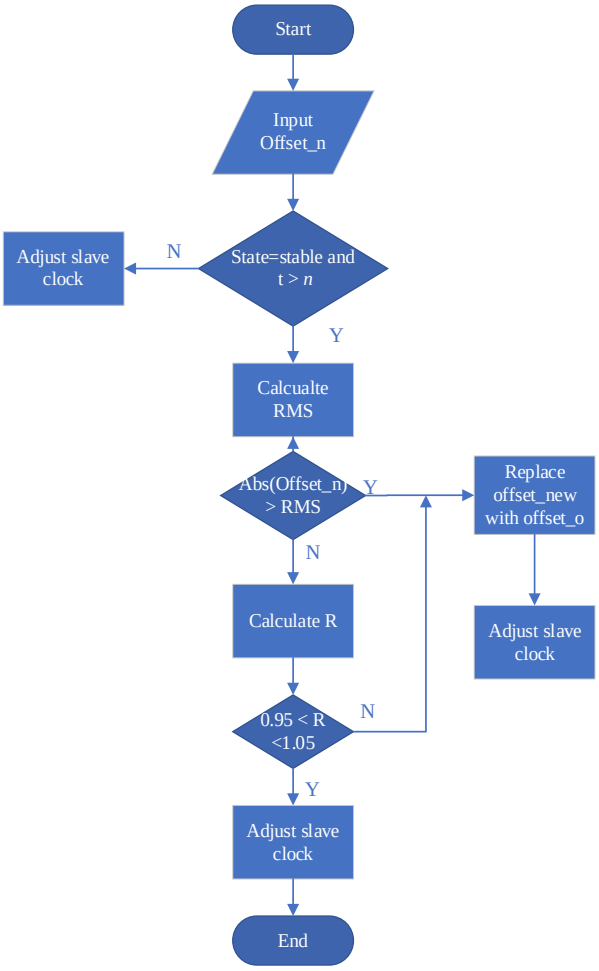

Fig. 4. Algorithm flowchart

$$RMS = \sqrt{\frac{\sum_{i=1}^{n} offset_i^2}{n}} \tag{7}$$

IV. Implementation and Test of PTP Synchronization System

A. Software Design

The Linux needs the support of kernel to run the IEEE1588 protocol. Linux mainly includes PTP subsystem and clock driver. The PTP subsystem mainly completes two tasks, one is to abstract the time-frequency adjustment operation of the PTP clock device, define a unified architecture and a kernel call interface. The other is to provide a PTP character device

interface to the application layer. The overall framework is shown in Fig. 5.

Implement the corresponding procedures for running the protocol in user space. In the kernel space, it is divided into three sub-layers. PTP character device interface and system calls interacts with the application, followed by the PTP subsystem, The bottom layer is the clock driver, which completes the clock device time-frequency adjustment operation.

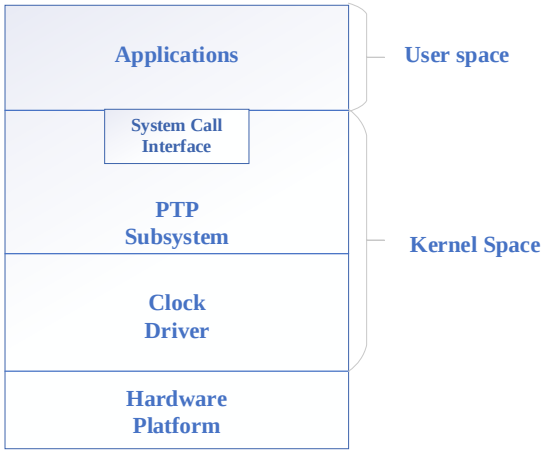

Fig. 5. Overall system architecture

The timestamp reading process is as follows: All operations are completed on the slave side. t2 and t3 are obtained through the slave. First, in the hardware of the network card, when sending and receiving data on the Ethernet, the timestamps have been written to the hardware FIFO. Secondly, the Ethernet driver reads the sending and receiving timing of the message from the FPGA FIFO. After obtaining the timestamp data, place it in the *hwtstamps* structure and submit it to the protocol stack for processing. Next, submit it to the socket interface of the application layer. Finally, at the application layer, call *recvmsg()* to read the message. So far the slave has obtained the data of t2 and t3.

The master puts the existing timestamp t1, whose length is 64 bits, into the *Follow_Up* message. When the *Follow_Up* message reaches the slave, because network packets have their corresponding identifiers (such as IP address and destination port, etc.), it is feasible to distinguish the type of the packet and associate it with its timestamp. Hence the slave can accurately obtain t1 from the packet at this moment. When *delay_resp* reaches the slave, the same as *Follow_up*, the slave can also get t4 information. So far, the timestamps of t1, t2, t3, and t4 have been accurately obtained. The calculations are performed on the PS side

When the PS-side operation is completed, the application layer directly calls *clock_adjtime()* to adjust the time and frequency. And the obtained frequency adjustment value ppb (frequency deviation ratio) is transmitted down to the drive layer. The drive layer completes the adjustment of the frequency.

B. Hardware Implementation

Figure 6 represents the block diagram of the Topology. Utilizing The integrated Gigabit Ethernet controller (GMAC) of the Zynq device, which contains timestamp unit completing the acquisition of the port time of the PTP packet . The bitstream will be downloaded to the FPGA, and the main design of the PL part is the real-time clock module and register module based on the adder. 'The Linux PTP Project' [16], an open source protocol, was modified and transplanted into the embedded Linux system. On the other hand, the 1588 drivers was developed, as it can allow the user space applications to access to hardware registers. IEEE 1588 IP Core interfaces with the PTP stack running on the ARM through the Advanced eXtensible Interface (AXI) bus. Hence the hardware clock has time and frequency modifiable functions and the ability to output PPS signals for precise testing of time synchronization accuracy. This structure allows users to adjust the registers in the IP core, increasing flexibility.

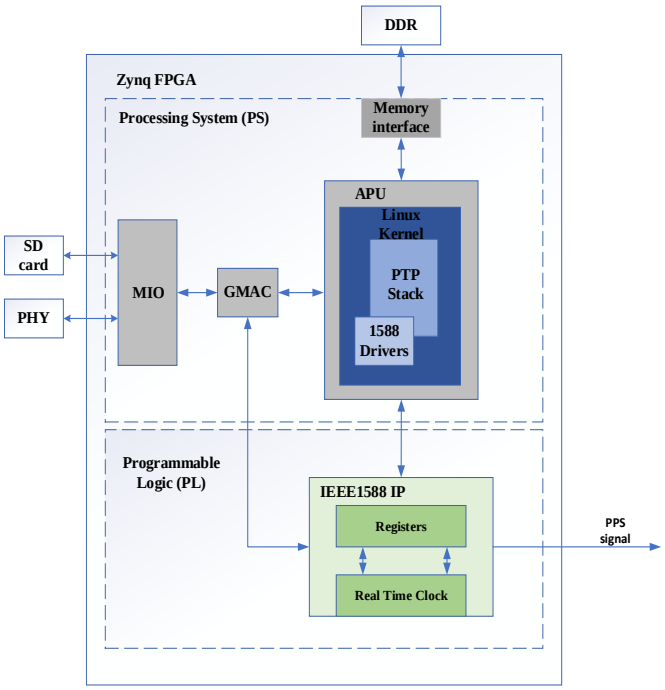

Fig. 6. The architecture of the system

Since Xilinx platform supports hardware time stamping, the stamping method used in this article is to stamp the on the MII (media independent interface) interface. Because the software timestamp is obtained by directly reading the system time through a system call, and the message is actually sent from the network card. The error is greater compared to time. Stamping here shields the effect of the layers above the PHY layer on delay. Therefore, higher clock synchronization accuracy can be obtained. The experiment master-slave machine uses the Zynq platform, the master chip is ZU7EV-2FFVC1156, and the slave chip is XCZU28DR-2FFVG1517E. The master and slave are directly connected through the optical fiber. The FPGA generates a local clock and records the timestamps for sending and receiving messages. The CPU implements the protocol calculation process, packet the message, and feeds back the time offset to the FPGA for adjustment. Connect to the PC with UART cable respectively, and the PC sends control commands. The PC system is Ubuntu 16.04, and the system on Zynq is Petalinux. After the algorithm is implemented, the test method is to derive the second pulse interface and measure the offset of the rising edge of the second pulse output by the oscilloscope. This test method of outputting the second pulse through

hardware has high accuracy. The diagrams of the setups are shown in Fig. 7. The pictures of these built setups can be seen in Fig. 8.

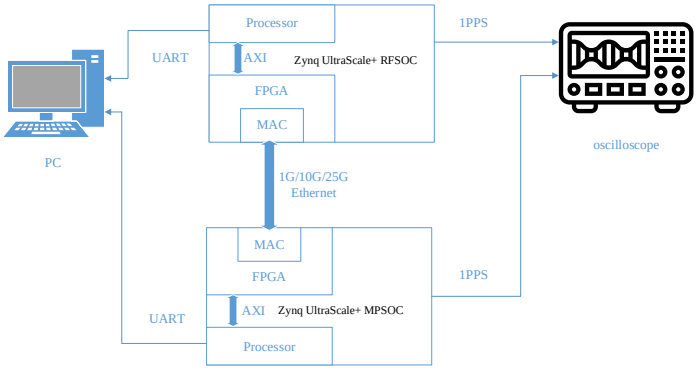

Fig. 7. Schemas of the experimental setups

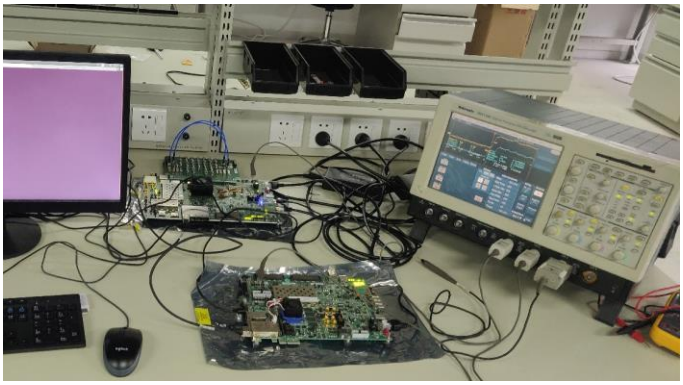

Fig. 8. Pictures of the experimental setups

C. *Implementation Results*

Through the above methods, an experimental environment is built. In Fig. 9, it can be seen that in the oscilloscope, the offset of the rising edges of the two waveforms is 27.77ns, indicating that the clock offset of the master clock and the slave clock is 27.77ns. This result meets the needs of most occasions.

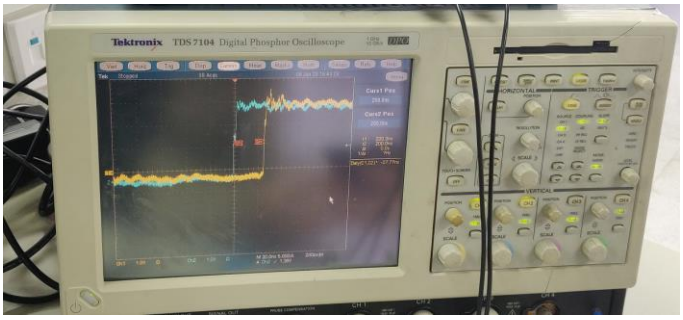

Fig. 9. The results of the test

Running the program, a system log of the clock offset will be generated on the PC. However, this value is less accurate than the second pulse output measurement offset, the effectiveness of the algorithm can also be verified by this.

Figure 10 shows the sample after the master-slave clock is stabilized. The three results are the comparison of hardware timestamping, software timestamping, and optimization algorithm used in PTP. It can be seen that the synchronization accuracy of the software timestamp method is on the order of microseconds, while the hardware timestamp and the optimized algorithm result are both on the order of nanoseconds.

Figure 11 shows the sampling after the master-slave clock is stabilized. Only compare hardware timestamping and optimized algorithms. The blue dot is the offset result of the master-slave clock using the conventional PTP algorithm. The red dot is the offset result of the optimized algorithm. The RMS of the traditional algorithm is 72.9ns, and the RMS of the optimized algorithm is 23.6ns. Therefore, the effectiveness of the optimized algorithm is better than the traditional algorithm.

It can be seen from Fig. 10 and Fig. 11. The time accuracy brought by the use of software timestamping can achieve the microsecond level. Sometimes can meet the system that is not particularly demanding on the time sequence. But it far from meeting the requirements of 5G mobile communication standards for time synchronization accuracy. Figure 12 shows the Cumulative Distribution Function of the offset. It can be seen that the proposed algorithm make the data more compact. The clock synchronization algorithm using hardware timestamping and the optimized clock synchronization algorithm both can meet the requirements of 5G communication standards for time synchronization accuracy. However, the optimized algorithm has better time synchronization accuracy than the clock synchronization algorithm using hardware timestamping in recent years. The optimized algorithm can meet the special service demands of some 5G communications.

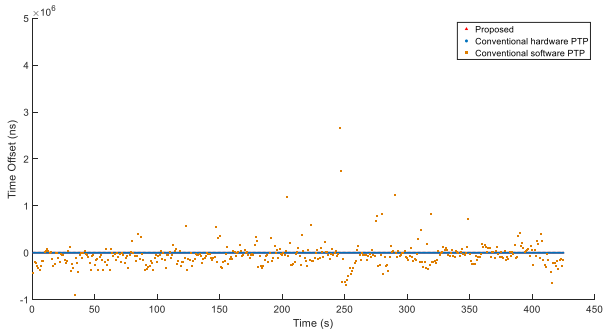

Fig. 10. Comparison of three methods

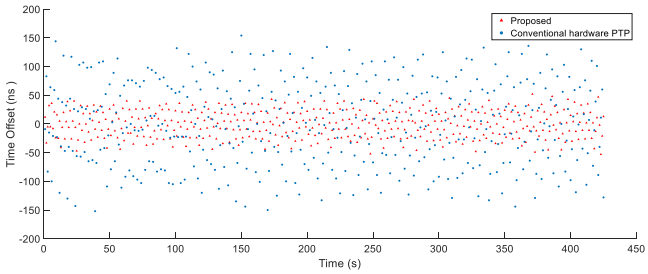

Fig. 11. Comparison of hardware timestamping and optimized algorithm

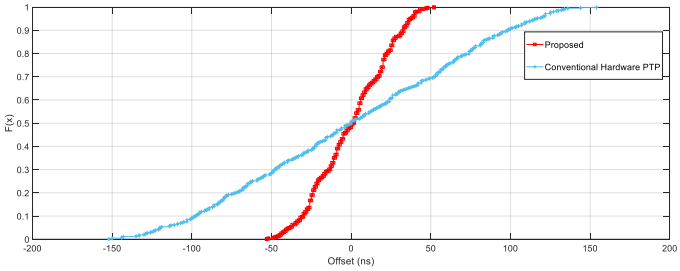

Fig. 12. The Cumulative Distribution Function of the offset

Figure 13 shows the complete process of running an optimized algorithm for master-slave clock synchronization. It can be seen that at the beginning, the offset of the master and slave clocks reached a maximum of 5899 nanosecond, but under the effect of the servo system and the filter, the data converged quickly. After 3 seconds, the data has converged. The slave clock is almost the same as the master clock. Since then the RMS of the clock offset is more than 20 ns, and there is no large jitter.

A histogram of the time offset is shown in Fig. 14. For the proposed PTP, it can be seen that most of the value were concentrated between ±40 nanoseconds. Hence, the proposed method is more stable and the sample distribution is more concentrated. For software PTP, the timer in the TSU cannot be used to directly schedule user-events in hardware. As a result, the accuracy of the clock is non-ideal, because of the software workload. Due to its low accuracy, only a few samples fall in the statistical interval.

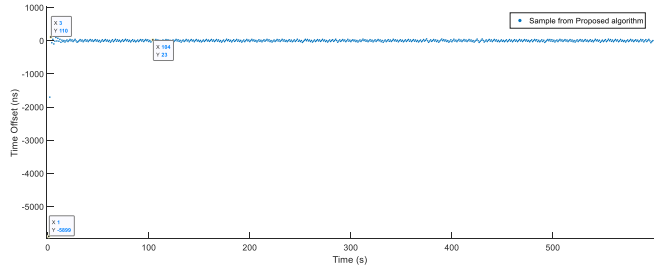

Fig. 13. The complete process of the optimized algorithm operation

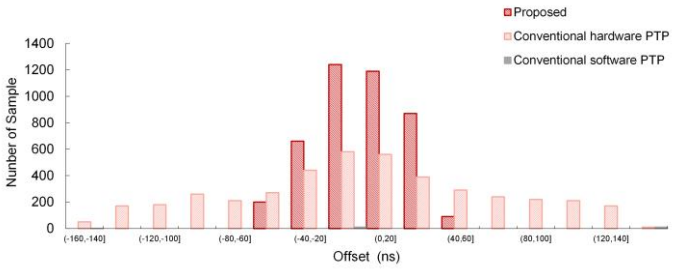

Fig. 14. The time offset histogram

V. Conclusion

For the traditional IEEE1588 clock synchronization scheme, it does not take into account the problem of link asymmetry in the network and packet delay changes. It is almost impossible for the upper and lower links to be symmetrical in the telecommunications network and the packet delay variations are always faced in the networks. But the traditional algorithm ignores the error caused by the above situation. In this paper, a hardware and software platform of FPGA+ARM is designed using hardware timestamping. Through experiments, an algorithm that introduces a twice filtering method for data filtering is proposed. The synchronization accuracy between the master and slave clocks is improved, and the time offset range within 40 nanoseconds when the system is stable. It is suitable for a variety of distributed systems, including the 5G open fronthaul network that require high synchronization accuracy.

References

[1] M. Dong, Z. Qiu, W. Pan, C. Chen, J. Zhang and D. Zhang, "The Design and Implementation of IEEE 1588v2 Clock Synchronization System by Generating Hardware Timestamps in MAC Layer," 2018 International Conference on Computer, Information and Telecommunication Systems (CITS), Colmar, 2018, pp. 1-5.

[2] Chavan A., Nagurvalli S., Jain M., Chaudhari S. (2018) Implementation of FPGA-Based Network Synchronization Using IEEE 1588 Precision Time Protocol (PTP). In: Sa P., Bakshi S., Hatzilygeroudis I., Sahoo M. (eds) Recent Findings in Intelligent Computing Techniques. Advances in Intelligent Systems and Computing, vol 708. Springer, Singapore.

[3] R. Exel, T. Bigler and T. Sauter, "Asymmetry Mitigation in IEEE 802.3 Ethernet for High-Accuracy Clock Syn chronization," in IEEE Transactions on Instrumentation and Measurement, vol. 63, no. 3, pp. 729-736, March 2014.

[4] W. Tseng, S. Siu, S. Lin and C. Liao, "Precise UTC dissemination through future telecom synchronization networks," 2015 Joint Conference of the IEEE International Frequency Control Symposium & the European Frequency and Time Forum, Denver, CO, 2015, pp. 696-699.

[5] O. Ronen and M. Lipinski, "Enhanced synchronization accuracy in IEEE1588," 2015 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), Beijing, 2015, pp. 76-81.

[6] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Sys tems," in IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002) , vol., no., pp.1-300, 24 July 2008.

[7] Eleftherios Kyriakakis, Jens Sparsø, and Martin Schoeberl. 2018. Hardware Assisted Clock Synchronization with the IEEE 1588-2008 Precision Time Protocol. In Proceedings of the 26th International Conference on Real-Time Networks and Systems (RTNS '18). Association for Computing Machinery, New York, NY, USA, 51–60.

[8] W.Jinqi, C.Hong, "Implementation of IEEE1588 Precision Clock Synchronization Protocol Based on ARM",2019,42(06):1527-1531.

[9] G. Giorgi and C. Narduzzi, "Performance Analysis of Kalman-Filter-Based Clock Synchronization in IEEE 1588 Networks," in *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 8, pp. 2902-2909, Aug. 2011.

[10] Lee S. An enhanced IEEE 1588 time synchronization algorithm for asymmetric communication link using block burst transmission[J]. IEEE communications letters, 2008, 12(9): 687-689.

[11] Chen, W., Sun, J., Zhang, L. et al. An implementation of IEEE 1588 protocol for IEEE 802.11 WLAN. Wireless Netw 21, 2069–2085 (2015).

[12] P. A. Crossley, H. Guo and Z. Ma, "Time synchronization for transmission substations using GPS and IEEE 1588," in CSEE Journal of Power and Energy Systems, vol. 2, no. 3, pp. 91-99, Sept. 2016.

[13] H. Guo and P. Crossley, "Design of a Time Synchronization System Based on GPS and IEEE 1588 for Transmission Substa tions," in IEEE Transactions on Power Delivery, vol. 32, no. 4, pp. 2091-2100, Aug. 2017.

[14] O. Seijo, I. Val, J. A. Lopez-Fernandez and M. Velez, "IEEE 1588 Clock Synchronization Performance over Time-Varying Wireless Channels," 2018 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), Geneva, 2018, pp. 1-6.

[15] S. Lee and C. Hong, "An Accuracy Enhanced IEEE 1588 Synchronization Protocol for Dynamically Changing and Asymmetric Wireless Links," IEEE Communications Letters, vol. 16, no. 2, pp. 190-192, February 2012.

[16] The Linux PTP Project. [Online]. Available:http://linuxptp.sourceforge.net/, accessed Dec. 2015.

[17] N. Moreira, J. Lázaro, U. Bidarte, J. Jimenez, and A.Astarloa, "On the Utilization of System-on-Chip Platformsto Achieve Nanosecond Synchronization Accuracies inSubstation Automation Systems."